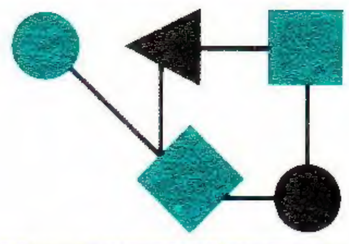
CONNEXIONS



The Interoperability Report

January 1988 Volume 2, No. 1

ConneXions - The Interoperability Report tracks current and emerging standards and technologies within the computer and communications industry.

In this issue:

ConneXions is published by Advanced Computing Environments, 21370 Vai Avenue, Cupertino, CA 95014, USA 408-996-2042.

ISSN 0894-5926

From the Editor

A great deal of effort is being put into protocol transition from the TCP/IP to OSI/ISO stack. This month Marshall Rose of The Wollongong Group describes another piece of the puzzle in his analysis of Transport-level "bridges". We will continue to bring you articles on OSI/ISO transition strategies in the following months.

The 2nd TCP/IP Interoperability Conference was held in Arlington, VA, December 1-4, 1987. On page 7 we bring you a conference report. This is followed by some pictorial highlights on pages 8 and 9.

Following his article in the December 1987 issue entitled "Mapping Internet Addresses to Ethernet Addresses", Doug Comer this month discusses reverse address mapping. The article is adapted from his new book *"Internetworking with TCP/IP Principles, Protocols, and Architecture"* which will be available next month.

As mentioned last month, Advanced Computing Environments will be sponsoring 9 tutorials in April. An overview of these tutorials can be found on page 14.

We would like to make you aware of some important work on TCP performance improvements which has been carried out by Van Jacobson of Lawrence Berkeley Laboratories. A new version of his experimental TCP for 4BSD UNIX systems is available for anonymous FTP from host **lbl-rtsg.arpa** (128.3.254.68 or 128.3.255.68). Transfer **xtcp.tar.Z**, a 50KB compressed tar archive. Uncompress the tar file and extract the README file from it for more information.

As we begin 1988, *ConneXions* has nearly 1200 subscribers. As editor, I would like to take this opportunity to thank all of our readers and contributors, and remind you that suggestions for topics are always welcome.

Transport-level "bridges" from TCP/IP to OSI/ISO

by Marshall Rose, The Wollongong Group

Introduction

Several approaches have been proposed regarding the peaceful co-existence of the TCP/IP and OSI/ISO suites. In this article, two such proposals are explained and reviewed for their technical merit.

The issue of co-existence and possible transition is an important one: There is a very large installed base of TCP/IP networks, and the rate of growth of networks using this technology is increasing. However, it is commonly acknowledged, often grudgingly, that protocols based on the Open Systems Interconnection (OSI) model as being promulgated by the International Organization for Standardization (ISO) will eventually achieve dominance and enjoy even greater success than TCP/IP. Although the exact point in time when TCP/IP will be eclipsed by OSI/ISO is still the center of much debate, it is clear that there will be a transition period, perhaps on the order of a decade, when both protocols will be in widespread use. Our interest is to examine how this co-existence can be leveraged by users in different communities.

Fundamental assumptions

Let us begin by stating the obvious: There are *many* TCP/IP networks today. Further, when OSI/ISO finally becomes a worthwhile alternative to TCP/IP, there will be many more TCP/IP networks than there are today! So, it should be clear from the start that we will have a sizable problem on our hands when OSI/ISO is "real". For reference, let's call this point in time, whenever it might be, the *Z-day*.

ISODE

Next, let's look at a not so obvious assumption: Even though there will be large number of TCP/IP networks on Z-day, these networks will actually offer a mix of services. That is, in addition to the standard TCP/IP applications such as FTP and SMTP, these TCP/IP networks will probably also offer ISO services as well, such as FTAM and X.400. One example of the technology which makes this possible is described in RFC 1006, "ISO Transport Services on top of the TCP". This RFC describes a simple method for making a TCP/IP-based network look like an ISO-based network. The technique is implemented in a software package called ISODE, *The ISO Development Environment*. [Ed. see *ConneXions*, Volume 1, No. 1, May 1987]. Using ISODE, applications such as the ISO File Transfer, Access and Management (FTAM) are already running over TCP/IP-based networks such as the DARPA/NSF Internet.

A fundamental assumption of the approach described herein is that it will be these ISO applications that will be used in both TCP/IP and OSI/ISO-based networks. That is, the same applications will be used in both communities; only the actual network and transport protocols (TCP/IP versus ISO TP4/CLNP) will differ between the two.

This assumption is not as far-fetched as it sounds: The ISO applications offer similar or superior functionality to their TCP/IP counterparts, which is not surprising considering that the ISO applications are much newer. Further, given the capability provided by RFC 1006, as implementations of the ISO applications are developed, they can be tested in both prototype ISO networks along with production TCP/IP networks.

Given these two assumptions, the problem of co-existence is straight-forward: Is there a way to run ISO applications between two communities, one of which is TCP/IP-based while the other is TP4/CLNP-based?

Transport-level gateways

In the research literature, there are several examples of both *encapsulation* and *translation* of various disjoint protocols. Encapsulation means that one protocol is carried within the other. Translation requires an interface between the two. For example, at the NATO SHAPE Technical Center, a protocol translator between the TCP and TP4 was built two years ago. To summarize this approach: Given two communities, one TCP/IP-based and the other TP4/CLNP-based, and given one host which is a member of both communities, one can build a transport gateway. This gateway "converts" TCP segments to TP4 TPDUs (transport protocol data units) by using some interesting algorithms and a fair amount of work. (Interested readers should consult the reference at the end of this article.) Although some power is lost from both TCP and TP4, the remaining functionality is quite substantial. (E.g., on the TP4 side, the loss of functionality should not prevent the ISO session provider from using the resulting transport service.) Figure 1 summarizes the relationships.

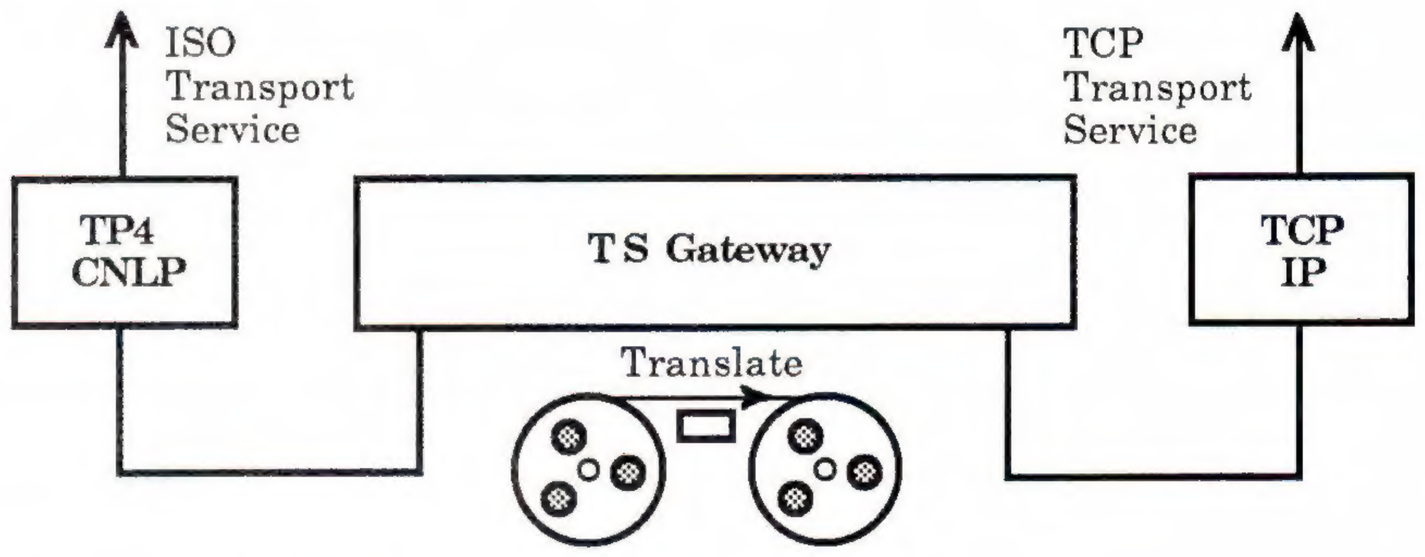


Figure 1: Transport Level Gateway

Middle-endpoint problem

From the point of view of connectivity, this approach has merit, as it permits us to join a TCP/IP-based network with a TP4/CLNP-based network. Unfortunately, there are several drawbacks. First, both TCP and TP4 are end-to-end services; that is, they offer a service between two endpoints. By introducing a transport gateway into the service, there are now really three endpoints! From a theoretical perspective, this is a terrible violation of the model for transport service. From the practical viewpoint, when a transport gateway is introduced it has a significant impact on reliability as there is now a new point of failure in the system. Since the transport gateway must maintain the state of the connections it is managing, whenever the gateway crashes, all connections are lost, even if the hosts are still running! This is the so-called *middle-endpoint* problem.

Different protocols

However, there is a more fundamental problem: What protocols do we run above the transport service on each host? Because different services are offered at each endpoint, *different* protocols must be run on each end! It is not even a simple matter of making an ISO application such as FTAM run over TCP, because whatever encapsulation scheme is used at the TCP-based host will also manifest itself at the TP4-based host, and this manifestation will be visible above the TP4 interface.

continued on next page

Transport-level "bridges" *(continued)*

This means that a third class of applications, which can operate simultaneously over both the TCP transport service and the ISO transport service, must be developed and run in this environment. Clearly this is unacceptable and perhaps not even implementable!

Transport-level bridges

The problem with the transport-level gateway approach is that it maintains the service interface of the underlying transport service at the service boundaries. Let's consider how we can still attain the connectivity benefits of the gateway while presenting a consistent service interface at both endpoints in order to achieve interoperability. With this approach, we agree on one transport service to offer at the endpoints, namely the ISO transport service. As mentioned earlier, RFC 1006 describes how to offer this service on TCP/IP-based networks. At this point, we have two disjoint communities offering the same service, as RFC 1006/TCP/IP and TP4/CLNP do not share an underlying protocol. In order to establish connectivity, we build a "bridge" which resides on one host which is a member of both the TCP/IP and TP4/CLNP-based communities. This bridge is a user of the transport services on both networks. As service primitives occur originating from one network, the bridge simply invokes the primitive on the other network. For example, when a transport connection is opened from the TP4/CLNP-based network, it opens a transport connection on the TCP/IP-based network.

Translate versus copy

It is critical to understand the fundamental differences in the approaches: with the gateway approach, the underlying protocol data units are *translated* (e.g., on receipt of a TCP SYN segment the gateway may generate a TP4 CR TPDU); in contrast, with the bridge approach, the higher-level service primitives are *copied* (e.g., when a connection request is received from one community, the bridge will propagate the connect request to the other community). That is, the bridge manages the state machines of the common transport service, whereas the gateway manages the state machines of transport protocols. The distinction is subtle but critical. Figure 2 summarizes the relationships.

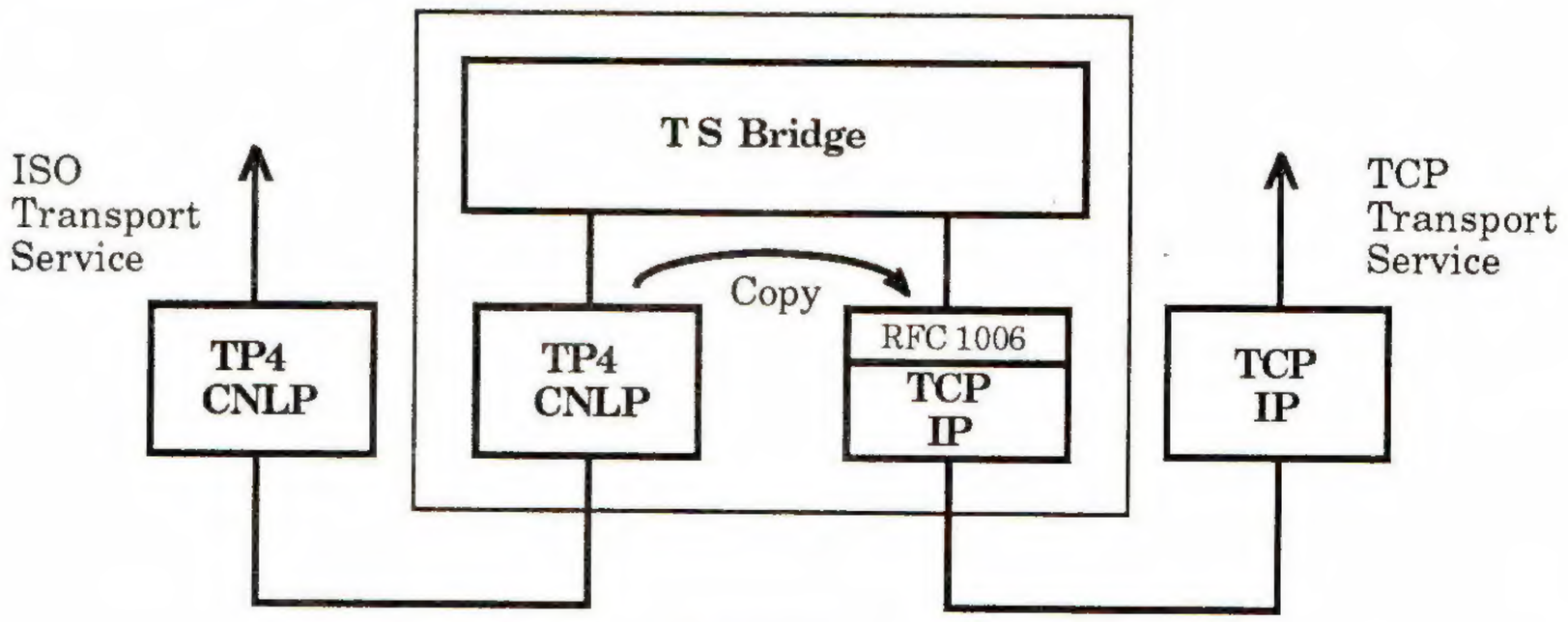


Figure 2: Transport Level Bridge

Of course, we still have the middle-endpoint problem with this approach, as the bridge must maintain the state of the connections it is managing. However, we are able to run one class of applications (namely ISO applications) on both networks and they do interoperate. Further, the transport-level bridge is much simpler to implement than a gateway (hence the arguable choice of this term).

The reason for this is that the services offered by the TCP and TP4 are relatively similar in comparison to the underlying protocols.

Transparent selection of bridges

Finally, let's consider how the use of directory services can actually make selection of the bridge transparent to both endpoints. Simply put, an application service can usually be referenced by a *presentation address*, which consists of a number of network addresses, and then three selectors (transport, session, and presentation). A network address identifies a particular location in the network, while the selectors (somehow) identify an entity at that location. For example, the service "gonzo-filestore" which identifies the entity providing the ISO file service on "gonzo" might have a corresponding presentation address like this:

```
network address:          CLNP 470005001700000800200040530 1
transport selector:       1
session selector:         "FTAM"
presentation selector:    null
```

In order to achieve transparent selection of the bridge, a slightly different entry is returned for hosts in the opposite community. The session and presentation selectors remain unchanged, but the network address and transport selector for the service are encoded as a new transport selector, then the network address of the bridge is substituted for the network address of the service. For example, if the service "gonzo-filestore" were really on a host in the opposite community, the entry might look like this instead:

```
network address:          bridge's network address
transport selector:       network address: CLNP 47...
                          transport selector: 1
session selector:         "FTAM"
presentation selector:    null
```

Providing that the same encoding scheme for transport selectors is used by the directory services database and the transport-level bridge, then operation of the bridge is satisfyingly straight-forward while the endpoints remain naive as to the presence of the bridge. Consider: When a host attempts to access a service, it asks directory services to provide the corresponding presentation address. The host then uses the network address of the bridge to establish a transport connection, thinking that it is contacting the host offering the service. The bridge decodes the actual network address and transport selector and establishes a new transport connection. The bridge then simply receives transport service primitives from one connection and invokes them on the other connection.

The premise here is that ISO-style addressing is sufficiently flexible to permit arbitrary encodings of information. (Although given the actual CLNP address given above, perhaps ISO-style addressing is *too* flexible!) Further with a little cooperation between the bridge and the directory services database, this extra level of indirection can be performed transparently to the software running at the endpoints.

Another problem solved

The transport-level bridge achieves connectivity across different underlying transport protocols which (somehow) provide the same transport service. In our example, the transport service offered is from ISO; one underlying protocol is the TP4, the other is RFC 1006 on top of the TCP.

continued on next page

Transport-level "bridges" *(continued)*

Readers familiar with this RFC will note that it merely specifies how to run ISO transport class 0 (TP0) on top of TCP. That is, RFC 1006 pretends that the TCP is a connection-oriented network service, and defines the rules for how TP0 uses this service.

In general then, the transport-level bridge described here is capable of interoperating between ISO TP0 and TP4.

TP0 or TP4

It turns out that it is likely that even in a pure ISO environment, the TP0 versus TP4 problem is going to exist! Broadly generalizing: In those countries with well-utilized public data networks, TP0/X.25 is the protocol combination of choice, while in other countries, TP4/CLNP is the protocol combination of choice. Naturally, a given connection can speak either TP0 or TP4 but not both. Hence, it is possible for a community to offer ISO transport services and not be interoperable with another ISO community.

The solution is to use exactly the kind of transport-level bridge described here! There is only one caveat: Because the services offered by TP0 and TP4 differ ever so slightly, the bridge has to "down negotiate" one facility (expedited data) and disregard one other facility (user data on connection) in order to make the two services identical. Fortunately, the ISO session protocol doesn't require either of these facilities for its operation.

Conclusions

Transport-level bridging is likely to provide a fairly clean method for interconnecting two communities offering similar transport services. Although not without its pitfalls, it appears to be a useful technology, not only during the co-existence of TCP/IP and OSI/ISO, but also as a harmonization method under the ISO umbrella.

Acknowledgements

Julian P. Onions of the Computer Science Department, University of Nottingham, England is responsible for the notion of TP0 transport bridges between X.25 and TCP-based networks. Such a TP0-bridge makes TCP-based hosts appear as X.25 hosts in a PDN. An RFC is currently under review describing this technique. The transport-level bridge described herein is a modification of Julian's original idea: The underlying network service is considered unimportant, and the transport class is permitted to vary.

References

"Conversion Between the TCP and ISO Transport Protocols as a Method of Achieving Interoperability Between Data Communication Systems", Inge Groenbaek, IEEE Journal on Selected Areas in Communication,Volume SAC-4, Number 2, March, 1986

"ISO Transport Services on top of the TCP", Marshall T. Rose and Dwight E. Cass, RFC 1006, May, 1987

MARSHALL T. ROSE is a Principal Software Engineer at The Wollongong Group. He is the principal implementor of the ISO Development Environment (ISODE), an openly available implementation of the upper layers of the ISO protocol stack. He was co-author of RFC 1006 (ISO Transport Services on top of the TCP), and was a member of the IFIP working group committee whose efforts led to RFC 987 (Mapping between X.400 and RFC 822). He is currently an advisor to the National Science Foundation, serving on its Network Technical Advisory Group. He is also an adjunct Assistant Professor at the University of Delaware. Rose received the Ph.D. degree in Information and Computer Science from the University of California, Irvine, in 1984.

Conference Report

The 2nd TCP/IP Interoperability Conference was held in Arlington, VA from December 1st through 4th, 1987. The conference attracted close to 900 attendees with about 1/3 representing vendors, 1/3 university users and 1/3 commercial users.

Tutorials

Five tutorials ranging from "Introduction to TCP/IP" to "Advanced Internet Topics" were offered on the fist day. More than 600 people attended these tutorials. Some of the tutorials will be given again in our April 1988 tutorial series (see page 14 for more details).

Network Management meeting

On December 1st an open meeting was held to discuss the status of Network Management. Presentations were given by the following groups:

GWMON (Gateway Management) Working Group of the Internet Engineering Task Force (IETF). GWMON is chaired by Craig Partridge of BBN Laboratories, and is the source of the High-Level Entity Management System (HEMS) proposal, documented in RFC 1021-1024 and recently implemented.

NETMAN (Network Management) Working Group of the IETF. Chaired by Lee LaBarre of Mitre Corp. This group is working on fashioning their own network management proposal, based partly on work done by the HEMS group.

SGMP (Simple Gateway Monitoring Protocol). Affiliated with the IETF, but not a working group (for odd historical reasons). SGMP is Documented in RFC 1028. Several test implementations exist.

The NETWORK ADMINISTRATION WORKING GROUP of The Network Computing Forum (NCF) is trying to develop higher level management applications/protocols. This is being led by Terry Hardgrave of the Software Productivity Consortium.

This meeting provided a useful opportunity to discuss the various proposals. It is hoped that such meetings can be arranged in the future, with the inclusion of yet more working groups such as those of MAP/TOP and NBS. Comprehensive Network Management is clearly an important missing piece in todays complex networking environments.

BOFs popular

In addition to 12 technical sessions and 20 Vendor Implementation Descriptions, the conference provided an opportunity for both pre-scheduled and spontaneous Birds of a Feather sessions (BOFs). More than 10 such sessions were held and proved very popular (even those scheduled at 7:30 am!).

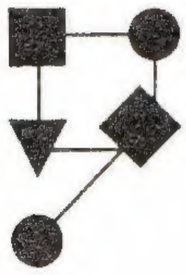
NetBIOS demo

On December 3, Bridge Communications, Excelan, Network General, Syntax and Ungermann-Bass successfully demonstrated compatibility to the NetBIOS over TCP/IP RFCs in a multi-vendor demonstration.

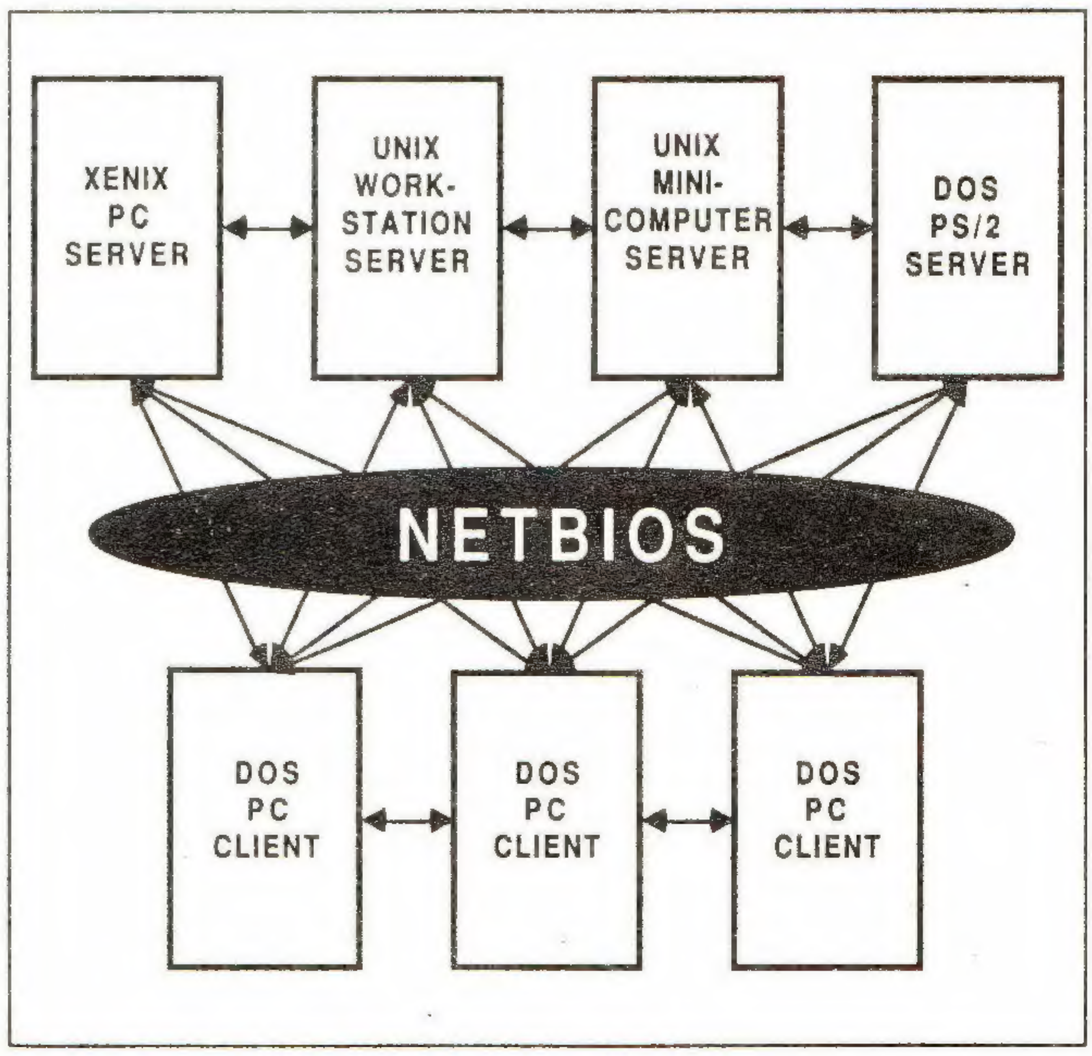
Exhibits next time

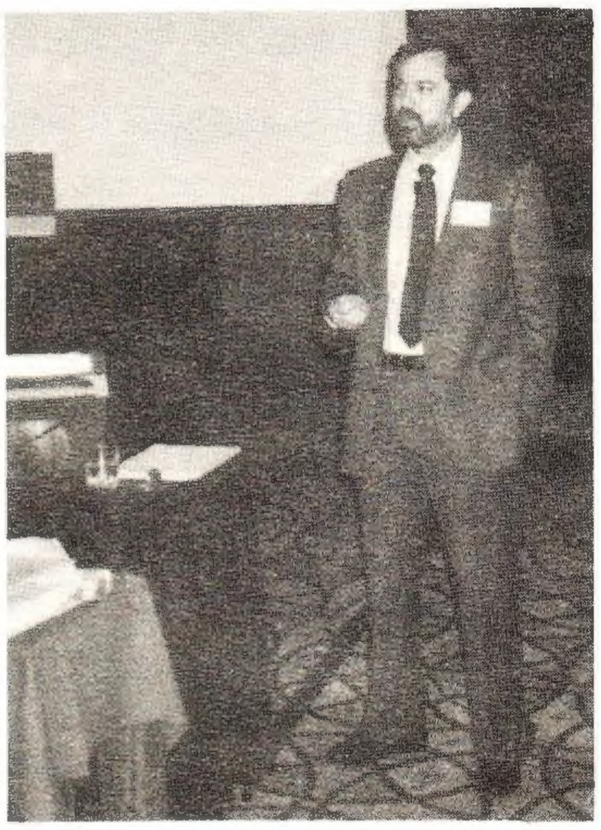
The 3rd TCP/IP Interoperability Conference will be held September 26 - 30, 1988 at the Santa Clara Convention Center. It will feature the first TCP/IP Exhibition & Solutions Showcase.

Conference Highlights

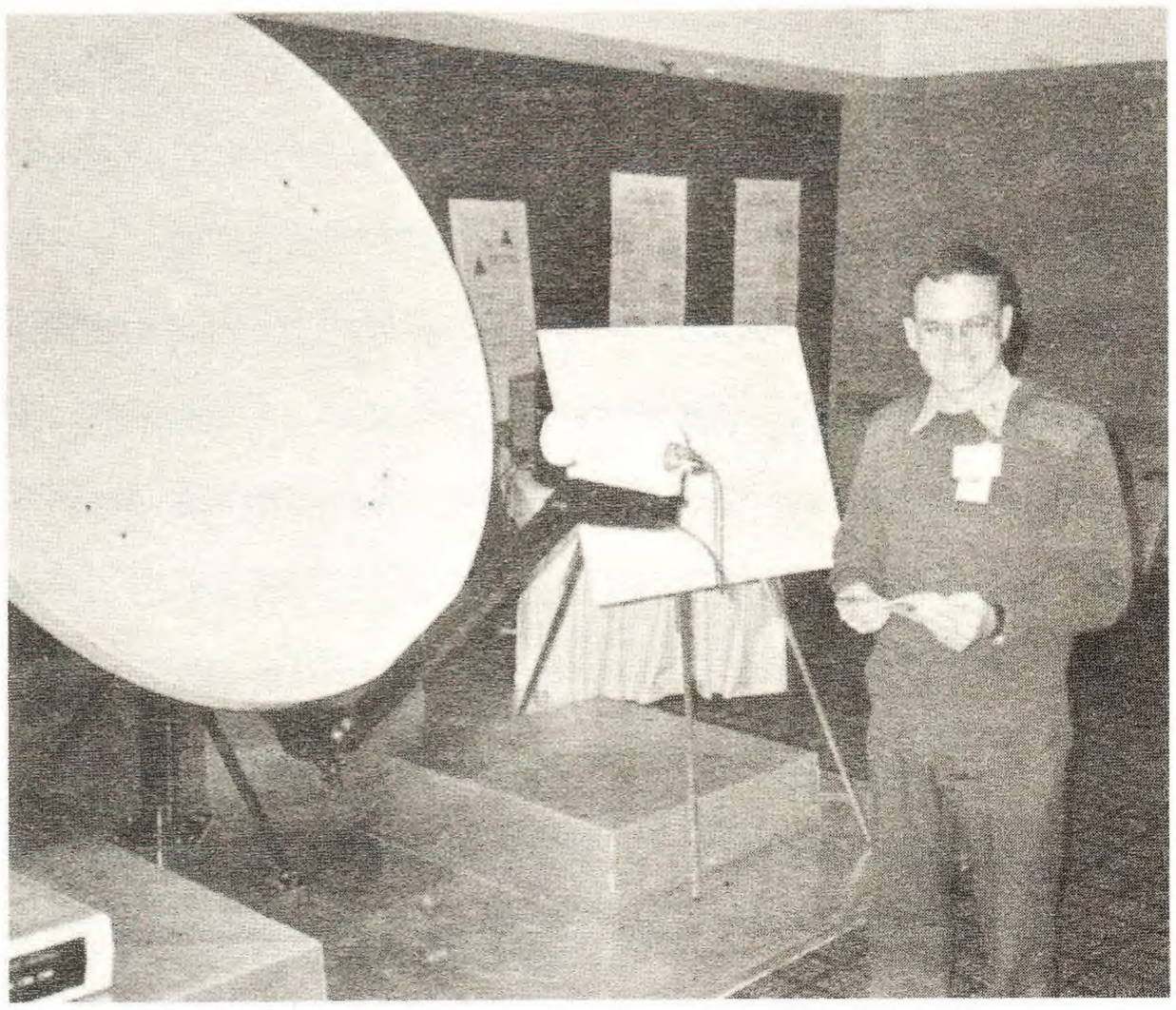


Above left: *Close to 900 attendees.* ***Above right:*** *Barry Leiner, RIACS with Jon Postel, USC-ISI.* ***Below right:*** *NetBIOS Compatibility Demonstration.* ***Below left:*** *Conference Reception.*





Above left: *Wendy Gibson, Helene Tannor, and Sara Tietz of Advanced Computing Environments.* ***Above right:*** *Bob Braden, USC-ISI: "What's New in Applications?".* ***Far right:*** *Tutorial: "Micros in TCP Land" by Dave Crocker, The Wollongong Group.* ***Below right:*** *Lt. Col. Mike Tallent in the DDN Information Center.* ***Below left:*** *SGMP authors Jeff Case, University of Tennessee and Marty Schoffstall, NYSERNET: "That Dog Will Hunt!"*



Determining an Internet Address at Startup

by Douglas Comer, Purdue University

If you recall from last month's article on the Address Resolution Protocol, machines on the Internet each have an assigned 32-bit Internet address that is independent of the machine's underlying physical network address. Although application programs always use the Internet address when specifying a destination, hosts and gateways must use physical addresses to transmit datagrams across underlying networks. Usually, a machine's Internet address is kept on its secondary storage, where the operating system finds it at startup. The question arises, "How does a diskless machine, one without access to secondary storage, determine its Internet address?" The problem is critical for diskless workstations that use Internet addresses to communicate with a file server.

This article discusses the most common way diskless machines determine their Internet addresses. It involves using the network to contact a server that can supply the needed address. The procedure sounds paradoxical: a machine communicates with a remote server over a network to obtain an address used for network communication.

The paradox is only imagined because the machine *does* know how to communicate. It can use its physical address to communicate over a single network. Thus, the machine must resort to physical network addressing temporarily in the same way that operating systems use physical memory addressing to set up page tables that allow virtual addressing. Once the machine knows its Internet address, it can communicate across the Internet.

Address servers

The idea behind finding an Internet address is simple: the diskless machine sends a request to a server and waits until the server sends a response. We assume the server has a disk where it keeps a database of Internet addresses. In the request, the machine needing to know its an Internet must uniquely identify itself so the server can look up the correct Internet address and send a reply. Both the machine that issues the request and the machine with the server use physical network addresses during the brief communication. How does the diskless machine know the physical address of the server? Usually, it does not - it simply broadcasts the request to all machines on the local network. One or more servers respond.

When the diskless machine broadcasts its request, it must uniquely identify itself. To keep the operating system image independent of the physical machine, the unique identification should not be compiled into the system image. What information can the operating system broadcast that will uniquely identify the machine on which it is executing? Any hardware identification suffices (e.g., the CPU serial number), but such information may be difficult to obtain, and the length or format may vary among CPUs.

The Internet designers realized that there is another piece of uniquely identifying information readily available, namely, the machine's physical network address. Using the physical address as a unique identification has two advantages. Because a host obtains its physical addresses from the network interface hardware, such addresses are always available and do not have to be bound into the operating system image.

Because the identifying information depends on the network and not on the CPU vendor or model, all machines on a given network will supply uniform, unique identifiers. Thus, the problem becomes the reverse of address resolution: given a physical network address, a server must map it into an Internet address.

RARP

The protocol diskless machines use to communicate with a server that can supply their Internet addresses is called the *Reverse Address Resolution Protocol* (RARP). It is adapted from the ARP protocol discussed last month and uses the same message format, as Figure 1 shows.

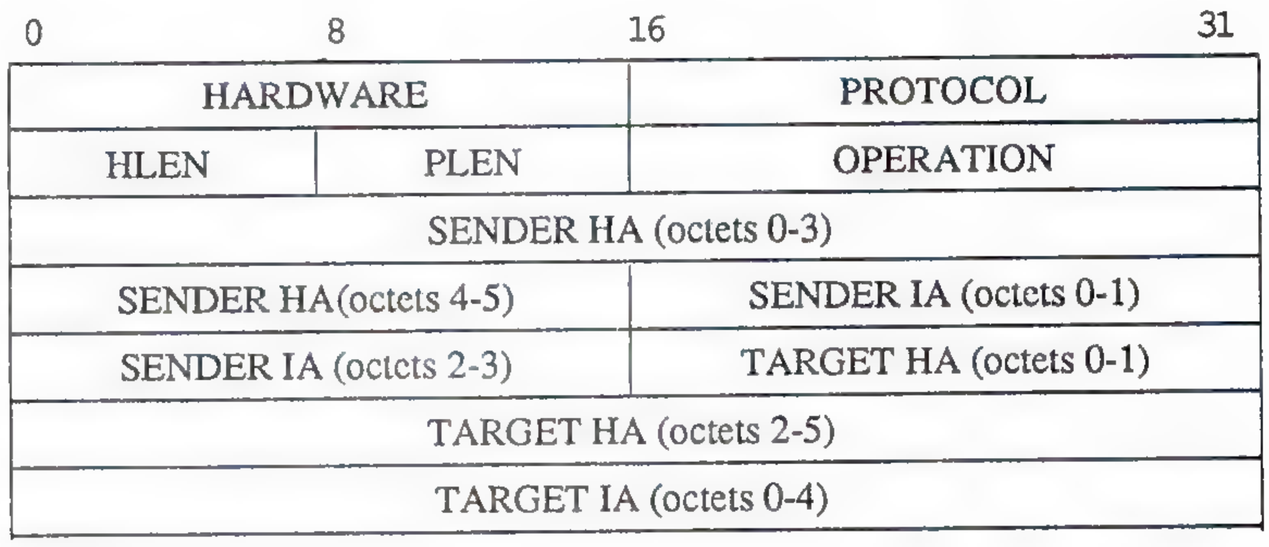


Figure 1: The format of ARP/RARP messages used for Internet-to-Ethernet address resolution.

Field HARDWARE specifies a hardware interface type for which the sender seeks an answer; it is 1 for Ethernet. Field OPERATION specifies a RARP request (3) or RARP response (4).

Fields HLEN and PLEN allow RARP to be used with arbitrary physical networks and arbitrary protocol addresses because they specify the length of the physical hardware address and the length of the protocol address.

When making a request, the sender supplies its physical hardware address, in field SENDER HA and the target hardware address using field TARGET HA. A response carries both the target machine's hardware address as supplied in the request and the target machine's Internet address (in field TARGET IA), as supplied by the server.

Notice that the RARP protocol is quite general in that it allows a machine to request the Internet address of a third party as easily as its own. It also allows for multiple types of physical network addresses.

Encapsulation

Like an ARP message, a RARP message is sent from one machine to another encapsulated in the data portion of an Ethernet frame. An Ethernet frame carrying the RARP request has the usual preamble, Ethernet source and destination addresses, and packet type fields in front of the frame. The frame type contains a value that identifies the contents of the frame as a RARP request. The data portion of the frame contains the 28-octet RARP message.

continued one next page

Determining an Internet Address at Startup *(continued)*

Figure 2 demonstrates how a host uses RARP. The sender broadcasts a RARP request that specifies itself as the target machine and supplies its physical hardware address. All machines on the network receive the request, but only those authorized to supply the RARP service process the request and send a reply; such machines are known as *RARP servers*. For RARP to succeed, each physical network must contain at least one RARP server.

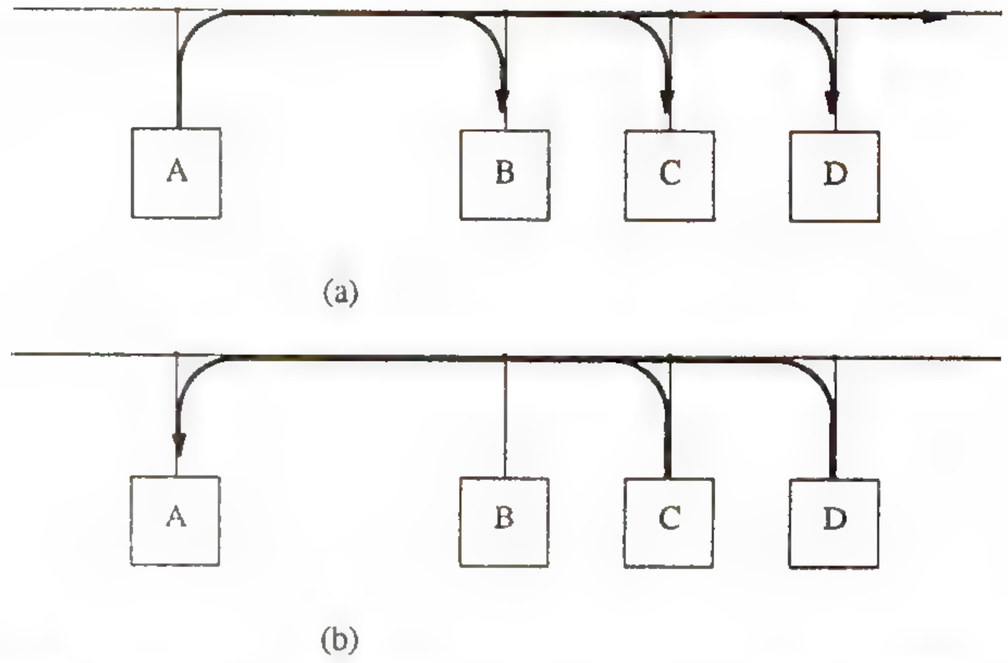


Figure 2: Example exchange using the RARP protocol. (a) host A broadcasts a RARP request specifying itself as a target, and (b) those machines authorized to supply the RARP service reply directly to A.

Servers answer requests by filling in the target protocol address field, changing the message type from *request* to *reply*, and sending the reply back directly to the machine making the request. The original machine receives replies from all RARP servers, even though only the first is needed.

Keep in mind that all communication between the host seeking its Internet address and the server supplying it must be carried out using only physical network addresses. Furthermore, the protocol allows a host to ask about an arbitrary target. Thus, the sender supplies its hardware address separate from the target hardware address, and the server is careful to send the reply using the sender's hardware address. On an Ethernet, having a field for the sender's hardware address may seem redundant because the information is also contained in the Ethernet frame header. However, not all Ethernet hardware provides the operating system with access to the physical frame header.

Timing RARP transactions

Like any network communication, RARP requests are susceptible to loss or corruption. Because RARP uses only the physical network, RARP software must handle timeout and retransmission. In general, RARP is used only on local area networks like the Ethernet, where the probability of failure is low. If a network has only one RARP server, however, that machine may not be able to handle the load, so packets may be dropped.

Many diskless machines rely on RARP to boot and may choose to retry indefinitely until they receive a response. Other implementations announce failure after only a few tries to avoid flooding the network with unnecessary broadcast traffic (e.g., in case the server is unavailable). On an Ethernet, network failure is less likely than overload at the server.

Making the RARP retransmit frequently may have the unwanted effect of flooding a congested server with more traffic. Thus, a large delay ensures that servers have ample time to satisfy the request and return an answer.

Primary and backup RARP Servers

The chief advantage of having several machines function as RARP servers is that it makes the system more reliable. If one server is down, or too heavily loaded to respond, another answers the request. Thus, it is highly likely that the service will be available. The chief disadvantage of using many servers is that when a machine broadcasts a RARP request, it can be overwhelmed by the responses.

How can the RARP service be arranged to keep it available and reliable without incurring the cost of multiple, simultaneous replies? There are at least two possibilities, and they both involve delaying responses. In the first solution, each machine that makes RARP requests is assigned a *primary server*. Under normal circumstances, only the machine's primary server responds to its RARP request. All nonprimary servers receive the request but merely record its arrival time. If the primary server is unavailable, the original machine will timeout waiting for a response and then rebroadcast the request. Whenever a nonprimary server receives a second copy of a RARP request within a short time of the first, it responds.

The second solution uses a similar scheme but attempts to avoid having all nonprimary servers transmit responses simultaneously. Each nonprimary machine that receives a request computes a random delay and then sends a response. Under normal circumstances, the primary server responds immediately, and successive responses are delayed so there is low probability that they arrive at the same time. When the primary server is unavailable, the requesting machine experiences a small delay before receiving a reply. By choosing delays carefully, the designer can guarantee that requesting machines do not timeout and rebroadcast before they receive an answer.

Summary

At system startup, a diskless machine must contact a server to find its Internet address before it can communicate on the Internet. We examined the RARP protocol that uses physical network addressing to obtain the machine's Internet address. The RARP mechanism supplies the target machine's physical hardware address to uniquely identify the processor and broadcasts the RARP request. Servers on the network receive the message, look up the mapping in a table (presumably from secondary storage), and reply to the sender.

Additional information

The details of RARP are given in Finlayson, et. al. [RFC 903]. Comer [1987] discusses an example implementation of RARP for the Xinu operating system. Often, a diskless machine needs to boot a new image at startup. To handle such cases, Croft and Gilmore propose an alternative to RARP called BOOTP [RFC 951]. Finlayson [RFC 906] describes workstation bootstrapping using the TFTP protocol.

continued on next page

Determining an Internet Address at Startup *(continued)*

References

Comer, D., Operating System Design Vol 2.: Internetworking with Xinu, Prentice-Hall, 1987.

Croft, B. and Gilmore, J. "Bootstrap Protocol," RFC 951, 1985

Finlayson, R. "Bootstrap Loading Using TFTP," RFC 906, 1984

Finlayson, R., Mann, T., Mogul, J., and Theimer, M. "A Reverse Address Resolution Protocol," RFC 903, 1984.

This article is taken from the forthcoming book by Douglas Comer:

Internetworking With TCP/IP Principles, Protocols, and Architecture , Prentice-Hall, 1987

[Ed. See also "Mapping Internet Addresses to Ethernet Addresses" by Douglas Comer in *ConneXions* Volume 1, No. 8, December 1987]

DOUGLAS COMER joined the faculty of the Computer Science Department at Purdue University after receiving a Ph.D. in Computer Science from Pennsylvania State University in 1976. He was principal investigator on the CSNET project, where he developed X25NET software. In addition to writing numerous papers on operating systems and networks, he developed the Xinu operating system and wrote two text books on operating system design. One of the books was written while Comer was on leave at Bell Laboratories. He is currently principal investigator of several network research projects, including the Cypress project, a member of the CSNET Executive Committee, and chairman of the CSNET Technical Committee, member of the Internet Activities Board (IAB), chairman of the Distributed Systems Architecture Board (DSAB), and editor for the journal *Software Practice and Experience*.

TCP/IP OSI/ISO Tutorials

Nine tutorials will be offered in Advanced Computing Environments' TCP/IP OSI/ISO Tutorials series April 25-27 in Arlington, VA. You will be receiving a full program in the mail shortly. In the meantime, here are overviews of the courses:

Intro to TCP/IP

Doug Comer's *Introduction to TCP/IP* is an excellent first course for anyone interested in designing or building products that use the Internet or TCP/IP protocols, anyone interested in specifying or choosing products that use or supply internet services, or anyone interested in understanding the principles that underlie the Internet technologies. It is especially pertinent for anyone who wants to read the literature (i.e., RFC documents) because it provides a conceptual overview that is extremely difficult to obtain elsewhere.

TCP/IP In-Depth

A two-day course entitled *TCP/IP In-Depth* will be given by Len Bosack of cisco Systems. The seminar addresses protocols and protocol models in today's business and technical context. The historical background of TCP/IP will be discussed, along with an overview of current problems, issues and standards.

TCP/IP for The VM Systems Programmer

Nick Gimbrone and Michael Hojnowski of Cornell University will offer a two day course on *TCP/IP for the VM Systems Programmer.* This is a unique opportunity to look at the IBM TCP/IP package and hardware offerings for VM. A brief review of what the TCP/IP protocols are and how they fit in the VM environment will be given. The features, installation, configuration, modification, operation and maintenance of such a system will be discussed.

Two OSI/ISO tutorials

Two courses on OSI/ISO will also be offered. Hal Folts of the Omnicom Institute will give a one-day introduction of the *OSI Reference Model and Protocols*. This course will provide a thorough understanding of the concepts and terminology of OSI, a working knowledge of the OSI architecture, an introduction to the seven layers of OSI protocols, and expert guidance in applying OSI to the evolution of distributed information systems.

Marshall Rose of The Wollongong Group will give a one day follow-on course entitled *Building Distributed Applications in an OSI Framework*. Marshall will give a thorough exposition of "The Applications Cookbook", a guide to building distributed applications using the ISO protocols.

Network security

Network Security is the topic of a one day course given by Stephen Walker and William Barker of Trusted Information Systems. This tutorial will cover all aspects of network security solutions and provide status on available and proposed technologies.

Microcomputer networking

David Crocker of The Wollongong Group will speak on *Microcomputer Networking with TCP/IP* in a one day tutorial. This course will discuss the nature of TCP/IP networking and the technical, economic, and organizational aspects of using microcomputers within larger TCP/IP networks.

Berkeley UNIX networking

Berkeley UNIX Networking will be covered in a two-day tutorial by Mike Karels of UC Berkeley. The course will offer a detailed look at the networking facilities provided by the 4.3BSD UNIX: networking protocol families and address formats; installation, configuration and management of networks; system layers, data structures, and interfaces between layers; the DoD TCP/IP protocol suite; routing protocols, options and strategy; and network servers and configurations.

Local Area Networks

Charles D. Brown of Complete Systems will offer a two-day course on *Local Area Networks* designed to provide the attendee with a thorough understanding of LAN technology. The program is designed to analyze LAN technology from both the perspective of the pieces that make up the LAN: cable, protocols, hardware, software etc., and from the big picture point of view. This course will allow attendees to make authoritative decisions about the direction of their Local Area Network environment. LAN selection issues and products are covered, giving users a set of guidelines to work with as they address their system needs.

For more information contact Advanced Computing Environments at 408-996-2042.

CONNEXIONS
21370 Vai Avenue
Cupertino, CA 95014

FIRST CLASS MAIL
U.S. POSTAGE
P A I D
CUPERTINO, CA
PERMIT NO. 782

CONNEXIONS

PUBLISHER Daniel C. Lynch

EDITOR Ole J. Jacobsen